

EXHIBIT B20

VirgilYP / peng Public

<> Code Issues Pull requests Actions Projects Security Insights

94451b22fc

...

peng / ext / hal / nxp / imx / drivers / ccm_imx7d.h



diegosueiro ext/hal/nxp/imx: Import the nxp imx7 freertos bsp ...

History

1 contributor

470 lines (432 sloc) 22.7 KB

...

```
1  /*
2  * Copyright (c) 2015-2016, Freescale Semiconductor, Inc.
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without modification,
6  * are permitted provided that the following conditions are met:
7  *
8  * o Redistributions of source code must retain the above copyright notice, this list
9  *   of conditions and the following disclaimer.
10 *
11 * o Redistributions in binary form must reproduce the above copyright notice, this
12 *   list of conditions and the following disclaimer in the documentation and/or
13 *   other materials provided with the distribution.
14 *
15 * o Neither the name of Freescale Semiconductor, Inc. nor the names of its
16 *   contributors may be used to endorse or promote products derived from this
17 *   software without specific prior written permission.
18 *
19 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
20 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
21 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
22 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
23 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
24 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
25 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
26 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
27 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
28 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29 */
```

```

30
31 #ifndef __CCM_IMX7D_H__
32 #define __CCM_IMX7D_H__
33
34 #include <stdint.h>
35 #include <stdbool.h>
36 #include <stddef.h>
37 #include <assert.h>
38 #include "device_imx.h"
39
40 /*!
41  * @addtogroup ccm_driver
42  * @{
43  */
44
45 /*****
46  * Definitions
47  *****/
48 #define CCM_REG_OFF(root, off) (*((volatile uint32_t *)((uint32_t)root + off)))
49 #define CCM_REG(root)          CCM_REG_OFF(root, 0)
50 #define CCM_REG_SET(root)      CCM_REG_OFF(root, 4)
51 #define CCM_REG_CLR(root)      CCM_REG_OFF(root, 8)
52
53 /*! @brief Root control names for root clock setting. */
54 enum _ccm_root_control
55 {
56     ccmRootM4      = (uint32_t)(&CCM_TARGET_ROOT1),    /*!< ARM Cortex-M4 Clock control name.*/
57     ccmRootAxi     = (uint32_t)(&CCM_TARGET_ROOT16),   /*!< AXI Clock control name.*/
58     ccmRootAhb     = (uint32_t)(&CCM_TARGET_ROOT32),   /*!< AHB Clock control name.*/
59     ccmRootIpg     = (uint32_t)(&CCM_TARGET_ROOT33),   /*!< IPG Clock control name.*/
60     ccmRootQspi    = (uint32_t)(&CCM_TARGET_ROOT85),   /*!< QSPI Clock control name.*/
61     ccmRootCan1    = (uint32_t)(&CCM_TARGET_ROOT89),   /*!< CAN1 Clock control name.*/
62     ccmRootCan2    = (uint32_t)(&CCM_TARGET_ROOT90),   /*!< CAN2 Clock control name.*/
63     ccmRootI2c1    = (uint32_t)(&CCM_TARGET_ROOT91),   /*!< I2C1 Clock control name.*/
64     ccmRootI2c2    = (uint32_t)(&CCM_TARGET_ROOT92),   /*!< I2C2 Clock control name.*/
65     ccmRootI2c3    = (uint32_t)(&CCM_TARGET_ROOT93),   /*!< I2C3 Clock control name.*/
66     ccmRootI2c4    = (uint32_t)(&CCM_TARGET_ROOT94),   /*!< I2C4 Clock control name.*/
67     ccmRootUart1   = (uint32_t)(&CCM_TARGET_ROOT95),   /*!< UART1 Clock control name.*/
68     ccmRootUart2   = (uint32_t)(&CCM_TARGET_ROOT96),   /*!< UART2 Clock control name.*/
69     ccmRootUart3   = (uint32_t)(&CCM_TARGET_ROOT97),   /*!< UART3 Clock control name.*/
70     ccmRootUart4   = (uint32_t)(&CCM_TARGET_ROOT98),   /*!< UART4 Clock control name.*/
71     ccmRootUart5   = (uint32_t)(&CCM_TARGET_ROOT99),   /*!< UART5 Clock control name.*/
72     ccmRootUart6   = (uint32_t)(&CCM_TARGET_ROOT100),  /*!< UART6 Clock control name.*/
73     ccmRootUart7   = (uint32_t)(&CCM_TARGET_ROOT101),  /*!< UART7 Clock control name.*/
74     ccmRootEcspi1  = (uint32_t)(&CCM_TARGET_ROOT102),  /*!< ECSP11 Clock control name.*/
75     ccmRootEcspi2  = (uint32_t)(&CCM_TARGET_ROOT103),  /*!< ECSP12 Clock control name.*/
76     ccmRootEcspi3  = (uint32_t)(&CCM_TARGET_ROOT104),  /*!< ECSP13 Clock control name.*/
77     ccmRootEcspi4  = (uint32_t)(&CCM_TARGET_ROOT105),  /*!< ECSP14 Clock control name.*/
78     ccmRootFtm1    = (uint32_t)(&CCM_TARGET_ROOT110),  /*!< FTM1 Clock control name.*/

```

#6934

```

79     ccmRootFtm2    = (uint32_t)(&CCM_TARGET_ROOT111), /*!< FTM2 Clock control name.*/
80     ccmRootGpt1    = (uint32_t)(&CCM_TARGET_ROOT114), /*!< GPT1 Clock control name.*/
81     ccmRootGpt2    = (uint32_t)(&CCM_TARGET_ROOT115), /*!< GPT2 Clock control name.*/
82     ccmRootGpt3    = (uint32_t)(&CCM_TARGET_ROOT116), /*!< GPT3 Clock control name.*/
83     ccmRootGpt4    = (uint32_t)(&CCM_TARGET_ROOT117), /*!< GPT4 Clock control name.*/
84     ccmRootWdog    = (uint32_t)(&CCM_TARGET_ROOT119), /*!< WDOG Clock control name.*/
85 };
86
87 /*! @brief Clock source enumeration for ARM Cortex-M4 core. */
88 enum _ccm_rootmux_m4
89 {
90     ccmRootmuxM4Osc24m    = 0U, /*!< ARM Cortex-M4 Clock from OSC 24M.*/
91     ccmRootmuxM4SysPllDiv2 = 1U, /*!< ARM Cortex-M4 Clock from SYSTEM PLL divided by 2.*/
92     ccmRootmuxM4EnetPll250m = 2U, /*!< ARM Cortex-M4 Clock from Ethernet PLL 250M.*/
93     ccmRootmuxM4SysPllPfd2 = 3U, /*!< ARM Cortex-M4 Clock from SYSTEM PLL PFD2.*/
94     ccmRootmuxM4DdrPllDiv2 = 4U, /*!< ARM Cortex-M4 Clock from DDR PLL divided by 2.*/
95     ccmRootmuxM4AudioPll   = 5U, /*!< ARM Cortex-M4 Clock from AUDIO PLL.*/
96     ccmRootmuxM4VideoPll   = 6U, /*!< ARM Cortex-M4 Clock from VIDEO PLL.*/
97     ccmRootmuxM4UsbPll     = 7U, /*!< ARM Cortex-M4 Clock from USB PLL.*/
98 };
99
100 /*! @brief Clock source enumeration for AXI bus. */
101 enum _ccm_rootmux_axi
102 {
103     ccmRootmuxAxiOsc24m    = 0U, /*!< AXI Clock from OSC 24M.*/
104     ccmRootmuxAxiSysPllPfd1 = 1U, /*!< AXI Clock from SYSTEM PLL PFD1.*/
105     ccmRootmuxAxiDdrPllDiv2 = 2U, /*!< AXI Clock DDR PLL divided by 2.*/
106     ccmRootmuxAxiEnetPll250m = 3U, /*!< AXI Clock Ethernet PLL 250M.*/
107     ccmRootmuxAxiSysPllPfd5 = 4U, /*!< AXI Clock SYSTEM PLL PFD5.*/
108     ccmRootmuxAxiAudioPll   = 5U, /*!< AXI Clock AUDIO PLL.*/
109     ccmRootmuxAxiVideoPll   = 6U, /*!< AXI Clock VIDEO PLL.*/
110     ccmRootmuxAxiSysPllPfd7 = 7U, /*!< AXI Clock SYSTEM PLL PFD7.*/
111 };
112
113 /*! @brief Clock source enumeration for AHB bus. */
114 enum _ccm_rootmux_ahb
115 {
116     ccmRootmuxAhbOsc24m    = 0U, /*!< AHB Clock from OSC 24M.*/
117     ccmRootmuxAhbSysPllPfd2 = 1U, /*!< AHB Clock from SYSTEM PLL PFD2.*/
118     ccmRootmuxAhbDdrPllDiv2 = 2U, /*!< AHB Clock from DDR PLL divided by 2.*/
119     ccmRootmuxAhbSysPllPfd0 = 3U, /*!< AHB Clock from SYSTEM PLL PFD0.*/
120     ccmRootmuxAhbEnetPll125m = 4U, /*!< AHB Clock from Ethernet PLL 125M.*/
121     ccmRootmuxAhbUsbPll     = 5U, /*!< AHB Clock from USB PLL.*/
122     ccmRootmuxAhbAudioPll   = 6U, /*!< AHB Clock from AUDIO PLL.*/
123     ccmRootmuxAhbVideoPll   = 7U, /*!< AHB Clock from VIDEO PLL.*/
124 };
125
126 /*! @brief Clock source enumeration for IPG bus. */
127 enum _ccm_rootmux_ipg

```

```

128 {
129     ccmRootmuxIpgAHB = 0U, /*!< IPG Clock from AHB Clock.*/
130 };
131
132 /*! @brief Clock source enumeration for QSPI peripheral. */
133 enum _ccm_rootmux_qspi
134 {
135     ccmRootmuxQspiOsc24m      = 0U, /*!< QSPI Clock from OSC 24M.*/
136     ccmRootmuxQspiSysPllPfd4  = 1U, /*!< QSPI Clock from SYSTEM PLL PFD4.*/
137     ccmRootmuxQspiDdrPllDiv2  = 2U, /*!< QSPI Clock from DDR PLL divided by 2.*/
138     ccmRootmuxQspiEnetPll500m = 3U, /*!< QSPI Clock from Ethernet PLL 500M.*/
139     ccmRootmuxQspiSysPllPfd3  = 4U, /*!< QSPI Clock from SYSTEM PLL PFD3.*/
140     ccmRootmuxQspiSysPllPfd2  = 5U, /*!< QSPI Clock from SYSTEM PLL PFD2.*/
141     ccmRootmuxQspiSysPllPfd6  = 6U, /*!< QSPI Clock from SYSTEM PLL PFD6.*/
142     ccmRootmuxQspiSysPllPfd7  = 7U, /*!< QSPI Clock from SYSTEM PLL PFD7.*/
143 };
144
145 /*! @brief Clock source enumeration for CAN peripheral. */
146 enum _ccm_rootmux_can
147 {
148     ccmRootmuxCanOsc24m      = 0U, /*!< CAN Clock from OSC 24M.*/
149     ccmRootmuxCanSysPllDiv4   = 1U, /*!< CAN Clock from SYSTEM PLL divided by 4.*/
150     ccmRootmuxCanDdrPllDiv2   = 2U, /*!< CAN Clock from SYSTEM PLL divided by 2.*/
151     ccmRootmuxCanSysPllDiv1   = 3U, /*!< CAN Clock from SYSTEM PLL divided by 1.*/
152     ccmRootmuxCanEnetPll40m   = 4U, /*!< CAN Clock from Ethernet PLL 40M.*/
153     ccmRootmuxCanUsbPll       = 5U, /*!< CAN Clock from USB PLL.*/
154     ccmRootmuxCanExtClk1      = 6U, /*!< CAN Clock from External Clock1.*/
155     ccmRootmuxCanExtClk34     = 7U, /*!< CAN Clock from External Clock34.*/
156 };
157
158 /*! @brief Clock source enumeration for ECSPI peripheral. */
159 enum _ccm_rootmux_ecspi
160 {
161     ccmRootmuxEcspiOsc24m     = 0U, /*!< ECSPI Clock from OSC 24M.*/
162     ccmRootmuxEcspiSysPllDiv2  = 1U, /*!< ECSPI Clock from SYSTEM PLL divided by 2.*/
163     ccmRootmuxEcspiEnetPll40m = 2U, /*!< ECSPI Clock from Ethernet PLL 40M.*/
164     ccmRootmuxEcspiSysPllDiv4  = 3U, /*!< ECSPI Clock from SYSTEM PLL divided by 4.*/
165     ccmRootmuxEcspiSysPllDiv1  = 4U, /*!< ECSPI Clock from SYSTEM PLL divided by 1.*/
166     ccmRootmuxEcspiSysPllPfd4  = 5U, /*!< ECSPI Clock from SYSTEM PLL PFD4.*/
167     ccmRootmuxEcspiEnetPll250m = 6U, /*!< ECSPI Clock from Ethernet PLL 250M.*/
168     ccmRootmuxEcspiUsbPll      = 7U, /*!< ECSPI Clock from USB PLL.*/
169 };
170
171 /*! @brief Clock source enumeration for I2C peripheral. */
172 enum _ccm_rootmux_i2c
173 {
174     ccmRootmuxI2cOsc24m       = 0U, /*!< I2C Clock from OSC 24M.*/
175     ccmRootmuxI2cSysPllDiv4    = 1U, /*!< I2C Clock from SYSTEM PLL divided by 4.*/
176     ccmRootmuxI2cEnetPll50m    = 2U, /*!< I2C Clock from Ethernet PLL 50M.*/

```

#6936

```

177     ccmRootmuxI2cDdrPllDiv2      = 3U, /*!< I2C Clock from DDR PLL divided by .*/
178     ccmRootmuxI2cAudioPll        = 4U, /*!< I2C Clock from AUDIO PLL.*/
179     ccmRootmuxI2cVideoPll        = 5U, /*!< I2C Clock from VIDEO PLL.*/
180     ccmRootmuxI2cUsbPll          = 6U, /*!< I2C Clock from USB PLL.*/
181     ccmRootmuxI2cSysPllPfd2Div2 = 7U, /*!< I2C Clock from SYSTEM PLL PFD2 divided by 2.*/
182 };
183
184 /*! @brief Clock source enumeration for UART peripheral. */
185 enum _ccm_rootmux_uart
186 {
187     ccmRootmuxUartOsc24m          = 0U, /*!< UART Clock from OSC 24M.*/
188     ccmRootmuxUartSysPllDiv2      = 1U, /*!< UART Clock from SYSTEM PLL divided by 2.*/
189     ccmRootmuxUartEnetPll40m     = 2U, /*!< UART Clock from Ethernet PLL 40M.*/
190     ccmRootmuxUartEnetPll100m    = 3U, /*!< UART Clock from Ethernet PLL 100M.*/
191     ccmRootmuxUartSysPllDiv1      = 4U, /*!< UART Clock from SYSTEM PLL divided by 1.*/
192     ccmRootmuxUartExtClk2         = 5U, /*!< UART Clock from External Clock 2.*/
193     ccmRootmuxUartExtClk34        = 6U, /*!< UART Clock from External Clock 34.*/
194     ccmRootmuxUartUsbPll          = 7U, /*!< UART Clock from USB PLL.*/
195 };
196
197 /*! @brief Clock source enumeration for FlexTimer peripheral. */
198 enum _ccm_rootmux_ftm
199 {
200     ccmRootmuxFtmOsc24m          = 0U, /*!< FTM Clock from OSC 24M.*/
201     ccmRootmuxFtmEnetPll100m     = 1U, /*!< FTM Clock from Ethernet PLL 100M.*/
202     ccmRootmuxFtmSysPllDiv4      = 2U, /*!< FTM Clock from SYSTEM PLL divided by 4.*/
203     ccmRootmuxFtmEnetPll40m     = 3U, /*!< FTM Clock from Ethernet PLL 40M.*/
204     ccmRootmuxFtmAudioPll        = 4U, /*!< FTM Clock from AUDIO PLL.*/
205     ccmRootmuxFtmExtClk3         = 5U, /*!< FTM Clock from External Clock 3.*/
206     ccmRootmuxFtmRef1m           = 6U, /*!< FTM Clock from Refernece Clock 1M.*/
207     ccmRootmuxFtmVideoPll        = 7U, /*!< FTM Clock from VIDEO PLL.*/
208 };
209
210 /*! @brief Clock source enumeration for GPT peripheral. */
211 enum _ccm_rootmux_gpt
212 {
213     ccmRootmuxGptOsc24m          = 0U, /*!< GPT Clock from OSC 24M.*/
214     ccmRootmuxGptEnetPll100m     = 1U, /*!< GPT Clock from Ethernet PLL 100M.*/
215     ccmRootmuxGptSysPllPfd0      = 2U, /*!< GPT Clock from SYSTEM PLL PFD0.*/
216     ccmRootmuxGptEnetPll40m     = 3U, /*!< GPT Clock from Ethernet PLL 40M.*/
217     ccmRootmuxGptVideoPll        = 4U, /*!< GPT Clock from VIDEO PLL.*/
218     ccmRootmuxGptRef1m           = 5U, /*!< GPT Clock from Refernece Clock 1M.*/
219     ccmRootmuxGptAudioPll        = 6U, /*!< GPT Clock from AUDIO PLL.*/
220     ccmRootmuxGptExtClk          = 7U, /*!< GPT Clock from External Clock.*/
221 };
222
223 /*! @brief Clock source enumeration for WDOG peripheral. */
224 enum _ccm_rootmux_wdog
225 {

```

#6937

```

226     ccmRootmuxWdogOsc24m      = 0U, /*!< WDOG Clock from OSC 24M.*/
227     ccmRootmuxWdogSysPl1Pfd2Div2 = 1U, /*!< WDOG Clock from SYSTEM PLL PFD2 divided by 2.*/
228     ccmRootmuxWdogSysPl1Div4     = 2U, /*!< WDOG Clock from SYSTEM PLL divided by 4.*/
229     ccmRootmuxWdogDdrPl1Div2     = 3U, /*!< WDOG Clock from DDR PLL divided by 2.*/
230     ccmRootmuxWdogEnetPl1125m    = 4U, /*!< WDOG Clock from Ethernet PLL 125M.*/
231     ccmRootmuxWdogUsbPl1         = 5U, /*!< WDOG Clock from USB PLL.*/
232     ccmRootmuxWdogRef1m          = 6U, /*!< WDOG Clock from Refernece Clock 1M.*/
233     ccmRootmuxWdogSysPl1Pfd1Div2 = 7U, /*!< WDOG Clock from SYSTEM PLL PFD1 divided by 2.*/
234 };
235
236 /*! @brief CCM PLL gate control. */
237 enum _ccm_pll_gate
238 {
239     ccmPl1GateCkil      = (uint32_t)(&CCM_PLL_CTRL0), /*!< Ckil PLL Gate.*/
240     ccmPl1GateArm       = (uint32_t)(&CCM_PLL_CTRL1), /*!< ARM PLL Gate.*/
241     ccmPl1GateArmDiv1   = (uint32_t)(&CCM_PLL_CTRL2), /*!< ARM PLL Div1 Gate.*/
242     ccmPl1GateDdr       = (uint32_t)(&CCM_PLL_CTRL3), /*!< DDR PLL Gate.*/
243     ccmPl1GateDdrDiv1   = (uint32_t)(&CCM_PLL_CTRL4), /*!< DDR PLL Div1 Gate.*/
244     ccmPl1GateDdrDiv2   = (uint32_t)(&CCM_PLL_CTRL5), /*!< DDR PLL Div2 Gate.*/
245     ccmPl1GateSys       = (uint32_t)(&CCM_PLL_CTRL6), /*!< SYSTEM PLL Gate.*/
246     ccmPl1GateSysDiv1   = (uint32_t)(&CCM_PLL_CTRL7), /*!< SYSTEM PLL Div1 Gate.*/
247     ccmPl1GateSysDiv2   = (uint32_t)(&CCM_PLL_CTRL8), /*!< SYSTEM PLL Div2 Gate.*/
248     ccmPl1GateSysDiv4   = (uint32_t)(&CCM_PLL_CTRL9), /*!< SYSTEM PLL Div4 Gate.*/
249     ccmPl1GatePfd0      = (uint32_t)(&CCM_PLL_CTRL10), /*!< PFD0 Gate.*/
250     ccmPl1GatePfd0Div2  = (uint32_t)(&CCM_PLL_CTRL11), /*!< PFD0 Div2 Gate.*/
251     ccmPl1GatePfd1      = (uint32_t)(&CCM_PLL_CTRL12), /*!< PFD1 Gate.*/
252     ccmPl1GatePfd1Div2  = (uint32_t)(&CCM_PLL_CTRL13), /*!< PFD1 Div2 Gate.*/
253     ccmPl1GatePfd2      = (uint32_t)(&CCM_PLL_CTRL14), /*!< PFD2 Gate.*/
254     ccmPl1GatePfd2Div2  = (uint32_t)(&CCM_PLL_CTRL15), /*!< PDF2 Div2.*/
255     ccmPl1GatePfd3      = (uint32_t)(&CCM_PLL_CTRL16), /*!< PDF3 Gate.*/
256     ccmPl1GatePfd4      = (uint32_t)(&CCM_PLL_CTRL17), /*!< PDF4 Gate.*/
257     ccmPl1GatePfd5      = (uint32_t)(&CCM_PLL_CTRL18), /*!< PDF5 Gate.*/
258     ccmPl1GatePfd6      = (uint32_t)(&CCM_PLL_CTRL19), /*!< PDF6 Gate.*/
259     ccmPl1GatePfd7      = (uint32_t)(&CCM_PLL_CTRL20), /*!< PDF7 Gate.*/
260     ccmPl1GateEnet       = (uint32_t)(&CCM_PLL_CTRL21), /*!< Ethernet PLL Gate.*/
261     ccmPl1GateEnet500m   = (uint32_t)(&CCM_PLL_CTRL22), /*!< Ethernet 500M PLL Gate.*/
262     ccmPl1GateEnet250m   = (uint32_t)(&CCM_PLL_CTRL23), /*!< Ethernet 250M PLL Gate.*/
263     ccmPl1GateEnet125m   = (uint32_t)(&CCM_PLL_CTRL24), /*!< Ethernet 125M PLL Gate.*/
264     ccmPl1GateEnet100m   = (uint32_t)(&CCM_PLL_CTRL25), /*!< Ethernet 100M PLL Gate.*/
265     ccmPl1GateEnet50m    = (uint32_t)(&CCM_PLL_CTRL26), /*!< Ethernet 50M PLL Gate.*/
266     ccmPl1GateEnet40m    = (uint32_t)(&CCM_PLL_CTRL27), /*!< Ethernet 40M PLL Gate.*/
267     ccmPl1GateEnet25m    = (uint32_t)(&CCM_PLL_CTRL28), /*!< Ethernet 25M PLL Gate.*/
268     ccmPl1GateAudio      = (uint32_t)(&CCM_PLL_CTRL29), /*!< AUDIO PLL Gate.*/
269     ccmPl1GateAudioDiv1  = (uint32_t)(&CCM_PLL_CTRL30), /*!< AUDIO PLL Div1 Gate.*/
270     ccmPl1GateVideo      = (uint32_t)(&CCM_PLL_CTRL31), /*!< VIDEO PLL Gate.*/
271     ccmPl1GateVideoDiv1  = (uint32_t)(&CCM_PLL_CTRL32), /*!< VIDEO PLL Div1 Gate.*/
272 };
273
274 /*! @brief CCM CCGR gate control. */

```



```

275  enum _ccm_ccgr_gate
276  {
277      ccmCcgrGateSimWakeup = (uint32_t)(&CCM_CCGR9),    /*!< Wakeup Mix Bus Clock Gate.*/
278      ccmCcgrGateIpmux1    = (uint32_t)(&CCM_CCGR10),   /*!< IOMUX1 Clock Gate.*/
279      ccmCcgrGateIpmux2    = (uint32_t)(&CCM_CCGR11),   /*!< IOMUX2 Clock Gate.*/
280      ccmCcgrGateIpmux3    = (uint32_t)(&CCM_CCGR12),   /*!< IPMUX3 Clock Gate.*/
281      ccmCcgrGateOcram     = (uint32_t)(&CCM_CCGR17),   /*!< OCRAM Clock Gate.*/
282      ccmCcgrGateOcramS    = (uint32_t)(&CCM_CCGR18),   /*!< OCRAM S Clock Gate.*/
283      ccmCcgrGateQspi      = (uint32_t)(&CCM_CCGR21),   /*!< QSPI Clock Gate.*/
284      ccmCcgrGateAdc       = (uint32_t)(&CCM_CCGR32),   /*!< ADC Clock Gate.*/
285      ccmCcgrGateRdc       = (uint32_t)(&CCM_CCGR38),   /*!< RDC Clock Gate.*/
286      ccmCcgrGateMu        = (uint32_t)(&CCM_CCGR39),   /*!< MU Clock Gate.*/
287      ccmCcgrGateSemaHs    = (uint32_t)(&CCM_CCGR40),   /*!< SEMA HS Clock Gate.*/
288      ccmCcgrGateSema1     = (uint32_t)(&CCM_CCGR64),   /*!< SEMA1 Clock Gate.*/
289      ccmCcgrGateSema2     = (uint32_t)(&CCM_CCGR65),   /*!< SEMA2 Clock Gate.*/
290      ccmCcgrGateCan1      = (uint32_t)(&CCM_CCGR116),  /*!< CAN1 Clock Gate.*/
291      ccmCcgrGateCan2      = (uint32_t)(&CCM_CCGR117),  /*!< CAN2 Clock Gate.*/
292      ccmCcgrGateEcspi1    = (uint32_t)(&CCM_CCGR120),  /*!< ECSPI1 Clock Gate.*/
293      ccmCcgrGateEcspi2    = (uint32_t)(&CCM_CCGR121),  /*!< ECSPI2 Clock Gate.*/
294      ccmCcgrGateEcspi3    = (uint32_t)(&CCM_CCGR122),  /*!< ECSPI3 Clock Gate.*/
295      ccmCcgrGateEcspi4    = (uint32_t)(&CCM_CCGR123),  /*!< ECSPI4 Clock Gate.*/
296      ccmCcgrGateGpt1      = (uint32_t)(&CCM_CCGR124),  /*!< GPT1 Clock Gate.*/
297      ccmCcgrGateGpt2      = (uint32_t)(&CCM_CCGR125),  /*!< GPT2 Clock Gate.*/
298      ccmCcgrGateGpt3      = (uint32_t)(&CCM_CCGR126),  /*!< GPT3 Clock Gate.*/
299      ccmCcgrGateGpt4      = (uint32_t)(&CCM_CCGR127),  /*!< GPT4 Clock Gate.*/
300      ccmCcgrGateI2c1      = (uint32_t)(&CCM_CCGR136),  /*!< I2C1 Clock Gate.*/
301      ccmCcgrGateI2c2      = (uint32_t)(&CCM_CCGR137),  /*!< I2C2 Clock Gate.*/
302      ccmCcgrGateI2c3      = (uint32_t)(&CCM_CCGR138),  /*!< I2C3 Clock Gate.*/
303      ccmCcgrGateI2c4      = (uint32_t)(&CCM_CCGR139),  /*!< I2C4 Clock Gate.*/
304      ccmCcgrGateUart1     = (uint32_t)(&CCM_CCGR148),  /*!< UART1 Clock Gate.*/
305      ccmCcgrGateUart2     = (uint32_t)(&CCM_CCGR149),  /*!< UART2 Clock Gate.*/
306      ccmCcgrGateUart3     = (uint32_t)(&CCM_CCGR150),  /*!< UART3 Clock Gate.*/
307      ccmCcgrGateUart4     = (uint32_t)(&CCM_CCGR151),  /*!< UART4 Clock Gate.*/
308      ccmCcgrGateUart5     = (uint32_t)(&CCM_CCGR152),  /*!< UART5 Clock Gate.*/
309      ccmCcgrGateUart6     = (uint32_t)(&CCM_CCGR153),  /*!< UART6 Clock Gate.*/
310      ccmCcgrGateUart7     = (uint32_t)(&CCM_CCGR154),  /*!< UART7 Clock Gate.*/
311      ccmCcgrGateWdog1     = (uint32_t)(&CCM_CCGR156),  /*!< WDOG1 Clock Gate.*/
312      ccmCcgrGateWdog2     = (uint32_t)(&CCM_CCGR157),  /*!< WDOG2 Clock Gate.*/
313      ccmCcgrGateWdog3     = (uint32_t)(&CCM_CCGR158),  /*!< WDOG3 Clock Gate.*/
314      ccmCcgrGateWdog4     = (uint32_t)(&CCM_CCGR159),  /*!< WDOG4 Clock Gate.*/
315      ccmCcgrGateGpio1     = (uint32_t)(&CCM_CCGR160),  /*!< GPIO1 Clock Gate.*/
316      ccmCcgrGateGpio2     = (uint32_t)(&CCM_CCGR161),  /*!< GPIO2 Clock Gate.*/
317      ccmCcgrGateGpio3     = (uint32_t)(&CCM_CCGR162),  /*!< GPIO3 Clock Gate.*/
318      ccmCcgrGateGpio4     = (uint32_t)(&CCM_CCGR163),  /*!< GPIO4 Clock Gate.*/
319      ccmCcgrGateGpio5     = (uint32_t)(&CCM_CCGR164),  /*!< GPIO5 Clock Gate.*/
320      ccmCcgrGateGpio6     = (uint32_t)(&CCM_CCGR165),  /*!< GPIO6 Clock Gate.*/
321      ccmCcgrGateGpio7     = (uint32_t)(&CCM_CCGR166),  /*!< GPIO7 Clock Gate.*/
322      ccmCcgrGateIomux     = (uint32_t)(&CCM_CCGR168),  /*!< IOMUX Clock Gate.*/
323      ccmCcgrGateIomuxLpsr = (uint32_t)(&CCM_CCGR169),  /*!< IOMUX LPSR Clock Gate.*/

```



```

324 };
325
326 /*! @brief CCM gate control value. */
327 enum _ccm_gate_value
328 {
329     ccmClockNotNeeded    = 0x0U,    /*!< Clock always disabled.*/
330     ccmClockNeededRun    = 0x1111U, /*!< Clock enabled when CPU is running.*/
331     ccmClockNeededRunWait = 0x2222U, /*!< Clock enabled when CPU is running or in WAIT mode.*/
332     ccmClockNeededAll    = 0x3333U, /*!< Clock always enabled.*/
333 };
334
335 /*****
336  * API
337  *****/
338
339 #if defined(__cplusplus)
340 extern "C" {
341 #endif
342
343 /*!
344  * @name CCM Root Setting
345  * @{
346  */
347
348 /*!
349  * @brief Set clock root mux
350  *
351  * @param base CCM base pointer.
352  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
353  * @param mux Root mux value (see @ref _ccm_rootmux_xxx enumeration)
354  */
355 static inline void CCM_SetRootMux(CCM_Type * base, uint32_t ccmRoot, uint32_t mux)
356 {
357     CCM_REG(ccmRoot) = (CCM_REG(ccmRoot) & (~CCM_TARGET_ROOT_MUX_MASK)) |
358                       CCM_TARGET_ROOT_MUX(mux);
359 }
360
361 /*!
362  * @brief Get clock root mux
363  *
364  * @param base CCM base pointer.
365  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
366  * @return root mux value (see @ref _ccm_rootmux_xxx enumeration)
367  */
368 static inline uint32_t CCM_GetRootMux(CCM_Type * base, uint32_t ccmRoot)
369 {
370     return (CCM_REG(ccmRoot) & CCM_TARGET_ROOT_MUX_MASK) >> CCM_TARGET_ROOT_MUX_SHIFT;
371 }
372

```

```

373  /*!
374  * @brief Enable clock root
375  *
376  * @param base CCM base pointer.
377  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
378  */
379  static inline void CCM_EnableRoot(CCM_Type * base, uint32_t ccmRoot)
380  {
381      CCM_REG_SET(ccmRoot) = CCM_TARGET_ROOT_SET_ENABLE_MASK;
382  }
383
384  /*!
385  * @brief Disable clock root
386  *
387  * @param base CCM base pointer.
388  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
389  */
390  static inline void CCM_DisableRoot(CCM_Type * base, uint32_t ccmRoot)
391  {
392      CCM_REG_CLR(ccmRoot) = CCM_TARGET_ROOT_CLR_ENABLE_MASK;
393  }
394
395  /*!
396  * @brief Check whether clock root is enabled
397  *
398  * @param base CCM base pointer.
399  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
400  * @return CCM root enabled or not.
401  *         - true: Clock root is enabled.
402  *         - false: Clock root is disabled.
403  */
404  static inline bool CCM_IsRootEnabled(CCM_Type * base, uint32_t ccmRoot)
405  {
406      return (bool)(CCM_REG(ccmRoot) & CCM_TARGET_ROOT_ENABLE_MASK);
407  }
408
409  /*!
410  * @brief Set root clock divider
411  *
412  * @param base CCM base pointer.
413  * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
414  * @param pre Pre divider value (0-7, divider=n+1)
415  * @param post Post divider value (0-63, divider=n+1)
416  */
417  void CCM_SetRootDivider(CCM_Type * base, uint32_t ccmRoot, uint32_t pre, uint32_t post);
418
419  /*!
420  * @brief Get root clock divider
421  *

```

```

422 * @param base CCM base pointer.
423 * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
424 * @param pre Pointer to pre divider value store address
425 * @param post Pointer to post divider value store address
426 */
427 void CCM_GetRootDivider(CCM_Type * base, uint32_t ccmRoot, uint32_t *pre, uint32_t *post);
428
429 /*!
430 * @brief Update clock root in one step, for dynamical clock switching
431 *
432 * @param base CCM base pointer.
433 * @param ccmRoot Root control (see @ref _ccm_root_control enumeration)
434 * @param root mux value (see @ref _ccm_rootmux_xxx enumeration)
435 * @param pre Pre divider value (0-7, divider=n+1)
436 * @param post Post divider value (0-63, divider=n+1)
437 */
438 void CCM_UpdateRoot(CCM_Type * base, uint32_t ccmRoot, uint32_t mux, uint32_t pre, uint32_t post);
439
440 /*@}*/
441
442 /*!
443 * @name CCM Gate Control
444 * @{
445 */
446
447 /*!
448 * @brief Set PLL or CCGR gate control
449 *
450 * @param base CCM base pointer.
451 * @param ccmGate Gate control (see @ref _ccm_pll_gate and @ref _ccm_ccgr_gate enumeration)
452 * @param control Gate control value (see @ref _ccm_gate_value)
453 */
454 static inline void CCM_ControlGate(CCM_Type * base, uint32_t ccmGate, uint32_t control)
455 {
456     CCM_REG(ccmGate) = control;
457 }
458
459 /*@}*/
460
461 #if defined(__cplusplus)
462 }
463 #endif
464
465 /*! @}*/
466
467 #endif /* __CCM_IMX7D_H__ */
468 /*****
469 * EOF
470 *****/

```

